التعابير النظامية

فى بيثون

من البداية و حتى الاحتراف

الجزء الأول

عبد التواب أحمد

التعابير النظامية في بيثون

من البداية وحتى الاحتراف

(الجزء الأول)

إعداد

عبد التواب أحمد

بسم الرحمن الرحيم

تعتبر بيثون من أسهل لغات البرمجة و أعظمها نفعا و انتشارا بين المبرمجين، كما أن تطبيقاتها مع هذه الفضائل تعد من أهم التطبيقات و خاصة تطبيقاتها فى مجال البحث العلمى و الاختراق و تحليل المعلومات، لذلك فأنا أنصح طلبة العلم الشرعى بتعلمها و كذلك الباحثين و المجاهدين وطلبة الكليات العلمية بل و حتى أطفال المسلمين و عوامهم، لكى تعم الاستفادة من هذه اللغة شديدة السهولة عظيمة الإمكانيات..

و إن شاء الله عز و جل أبدأ بتقديم هذه السلسة من الكتب (المترجمة "بتصرف" أو الملخصة أو المعتمدة على الجمع و الترتيب -على حسب الحاجة و التيسير-) فى لغة بيثون و ما يتعلق بها لأمتى الحبيبة أمة الإسلام سائلا الله عز و جل أن يوفقنى و أن ينفع بها الإسلام و المسلمين كافة و أن يتقبل منى و منكم إنه هو السميع العليم..

و الله عز و جل أسأل لى و لكم التوفيق و القبول و المغفرة و حسن الخاتمة و هو ثقتنا و رجاؤنا و مولانا فنعم المولى و نعم النصير.

أقدم هذا الكتاب لإخواني المسلمين و أسأل □ عز و جل أن ينفعهم به و أن يعينهم به على
نفع أمة الإسلام و رفع راية التوحيد ، و جزا الله خيرا من قام بنشره لتعم الفائدة و أسأل □
عز و جل لي و له القبول.

مقدمة عن التعابير النظامية

التعابير النظامية هى أنماط نصية تحدد الشكل الذى ينبغى أن تكون عليه سلسلة نصية ما (أو بمعنى آخر: سلسلة من الحروف "String").

و باستخدامها يمكن تنفيذ المهام التالية - بالإضافة إلى استعمالاتها أخرى -:

1- التحقق من أن البيانات المُدخلة متوافقة مع النموذج المُعطى، على سبيل المثال يمكننا التحقق من أن القيمة المدخلة عبر نموذج HTML هى بريد إلكترونى صالح أم لا.

2- كذلك يمكن البحث عن صيغة ما تكرر ظهورها فى قطعة نصية، و مثل ذلك التحقق من ظهور كلمة (affect) أو كلمة (effect) فى النص.

3- يمكن كذلك استخراج أجزاء معينة من النص، كاستخراج الرمز البريدى من العنوان.

4- كما يمكن استبدال كلمات من النص أينما ظهرت بكلمات أخرى، و مثاله استبدال كلمة ""Security أينما وجدت فى النص بكلمة "Islam".

5- تقسيم النصوص الكبيرة إلى قطع صغيرة، بحيث يبدأ التقسيم كلما وجدت فاصلة أو نقطة أو رمز "سطر جديد".

سنتعلم فى هذا الجزء أسس التعابير النظامية -من حيث هى لا من جهة لغة برمجية بعينها- بحيث نكون عند نهاية الجزء قد فهمنا جيدا كيفية عملها، و لكن طريقة استخدامها فى لغة بيثون ستكون هى موضوع الجزء الذى يليه و لذلك ستكون الأمثلة فى هذا الجزء عامة و نظرية.

تركيب التعبير النظامى (Regular Expression Syntax)

لاشك فى أنه ما من مبرمج مخضرم إلا و قد استعمل أحد أنواع التعابير النظامية ذات يوم، فعلى سبيل المثال، نجد أنه من الشائع استعمال كل من العلامتين (*) و (?) للعثور على الملفات فى لوحة تحكم نظام التشغيل، حيث تطابق علامة (?) رمزا مفردا واحدا له أى قيمة - قيمته غير محددة - فى اسم الملف.

فعلى سبيل المثال سيطابق هذا النمط (file?.html) جميع أسماء الملفات التالية و ما شابهها:

file1.html

filez.html

file2.html

files.html

و لكنه لن يطابق مثلا اسم الملف التالى: files56.html، و ذلك لأن النمط يقول: ابحث لى عن أى ملف اسمه يبدأ بكلمة file يليها رمز واحد مفرد بأى قيمة و ينتهى الاسم ب .html

و هذا على عكس ما يكون عند استعمال (*) إذ أن النمط ساعتها سيطابق اسم أى ملف يبدأ بكلمة file متبوعة بأى عدد من الرموز لها أى قيمة و منتهية ب .html، و هو ما يعنى أن نمط مثل file*.html سيطابق جميع ما يلى من أسماء الملفات:

file1.html

files56.html

filesmygreatjob.html

سنلاحظ فى التعبير السابق أن لدينا مكونين أساسيين اثنين و هما:

المكون الحرفى (literals) و يمثله: "file و html."

و المكون الرمزى (Metacharacters) و يمثله: "* و ?"

و مع أن نوع التعابير النظامية التى يتناولها هذا الكتاب بالشرح أكثر قوة من تلك الأنماط البسيطة التى يمكن أن نجدها فى نظام التشغيل إلا أن كلا النوعين يمكن أن يشتركا فى تعريف واحد و هو ما يلى:

التعبير النظامى: هو نمط (أو نموذج) نصى يتألف من رموز عادية مثل (الحروف من a إلى z أو الأرقام من 0 إلى 9) و رموز خاصة تعرف بل " metacharacters " ، وظيفة هذا النمط هو أنه يصف السلاسل النصية " strings " التى ينبغى مطابقتها عند تطبيق التعابير النظامية على نص من النصوص.

دعونا نرى تعبيرنا النظامى الأول و الذى سيطابق أى كلمة تبدأ ب:

⇢a

/⇢a\ w* /

حيث :

رمز المسافة (⇢) : حرفى Literal

حرف a : حرفى Literal

رمز \ : رمزى Metacharacter

رمز w : رمزى Metacharacter

رمز * : رمزى Metacharacter

شكل (التعابير النظامية) فى هذا الكتاب:

ستعرض التعابير النظامية فيما يلى من صفحات الكتاب متضمنة بين رمزين كهذين / / ، و يتم اتباع هذا فى نص الكتاب دون الأكواد الواردة فى الأمثلة.

و من جهة أخرى فلأنه من الصعب عد المسافات الفارغة فى التعبير النظامى حتى مع واجهة الخط أحادى المسافة، فقد تم الاستعاضة عن كل مسافة فارغة فى التعبير النظامى بالرمز ▹ لتيسير القراءة.

فى التعبير النظامى السابق:

/⇢a\ w* /

يمكن ترجمته بأنه يصف أى سلسلة نصية

تبدأ بحرف a بعد مسافة فارغة: (··→a)

يليه أى عدد من الرموز: (*)

يحمل أى قيمة حرفية أو رقمية بما فى ذلك "(underscore) الشرطة السفلية" : (\w)

سنشرح الجزء الرمزى Metacharacters من التعبير النظامى فيما بعد ، و لكن دعونا أولا نبدأ بفهم الجزء الحرفى منه Literals.

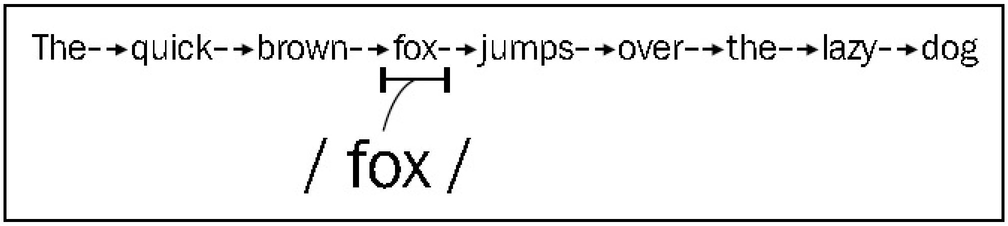
(Literals) الشق الحرفى من التعبير النظامى

تعتبر مطابقة [(Literals) الشق الحرفى من التعبير النظامى] أيسر أشكال مطابقة الأنماط فى التعابير النظامية ، إذ أن المطابقة ستنجح دائما كلما وجد هذا النمط الحرفى فى النص محل البحث.

فمثلا إذا طبقنا التعبير النظامى /fox/ عند البحث فى جملة:

The quick brown fox jumps over the lazy dog

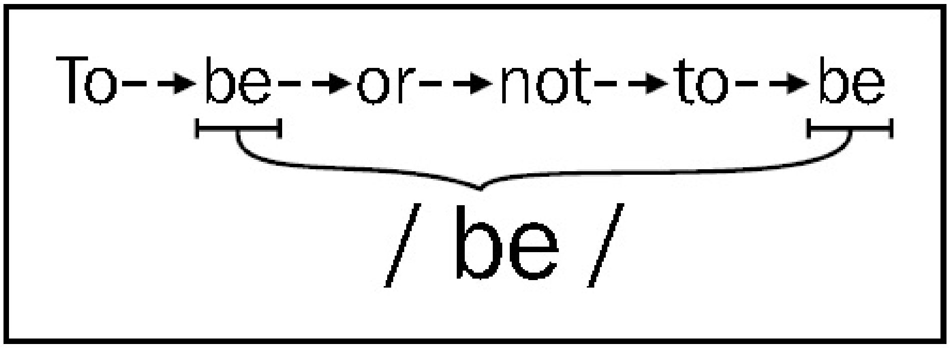
سنجد مطابقا واحدا لهذا التعبير.



(البحث باستعمال التعابير النظامية)

و يمكننا كذلك الحصول على نتائج بحث متعددة بدلا من نتيجة واحدة:

فمثلا إذا طبقنا التعبير النظامى /be/ على الجملة التالية: To be, or not to be سنحصل على نتائج متعددة للبحث بواسطة التعابير النظامية.



تعلمنا فيما مضى أن رموز ال Metacharacters يمكن أن تأتى جنبا إلى جنب مع ال Literals فى نفس التعبير النظامى، و بسبب هذا التصاحب بينهما يمكننا أن نجد أن بعض التعبيرات لا تعنى ما نقصده.

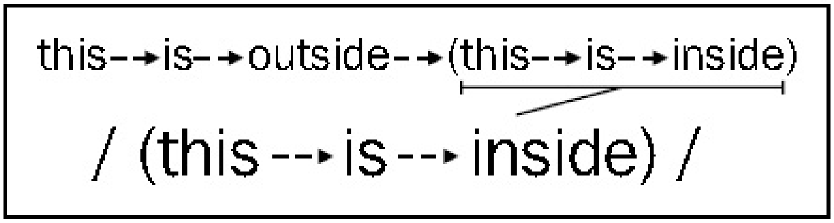
فعلى سبيل المثال إذا طبقنا التعبير النظامى:

/(this is inside)/

عند بحث الجملة:

this is outside (this is inside)

سنجد أن الأقواس غير متضمنة فى نتيجة البحث، و هذا يحدث بسبب أن الأقواس () من رموز ال Metacharacters مما يعنى بأن لها معنىً خاص.



لدينا ثلاث طرق لاستخدام رموز ال Metacharacters كما لو كانت رموز حرفية Literals و هذه الطرق هى:

1- تخطى رمز ال Metacharacter عن طريق سبقه برمز الخط المائل " backslash ".

2- استخدام نظام re.escape فى لغة بيثون لتخطى جميع الرموز غير (الأبجدية أو الرقمية) التى قد تظهر فى التعبير النظامى، و هذا ما سيتم شرحه فى الجزء الثانى عند الحديث عن التعابير النظامية فى لغة بيثون إن شاء الله عز و جل.

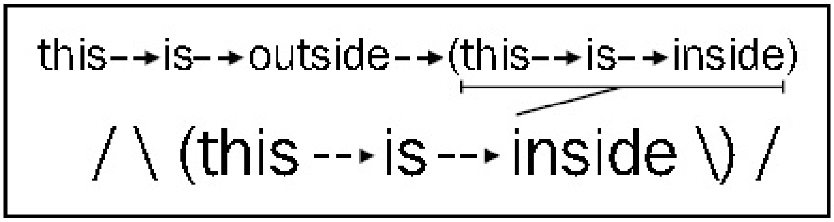
3- الحصر ضمن العلامتين \Q و \E (و هذه الطريقة غير متوفرة فى بيثون حتى الآن): و هى الطريقة الثالثة للحصر أو التضمين فى التعابير النظامية، و ذلك بحصر الجزء المراد بين الرمز \Q قبله و الرمز \E بعده.

و على أية حال فإنه يمكننا تحويل التعبير السابق المحتوى على الأقواس () و التى تم فهمها كرمز من رموز Metacharacters عن طريق لستخدام رمز الخط المائل backslash لكى يتم فهم الأقواس على أنها رمز حرفى-رقمى مع اجتنب معناها البرمجى مؤقتا فى هذا التعبير النظامى بعينه كالتالى:

```
/(\this is inside)\/
```

و بتطبيق التعبير على النص مرة أخرى نجد أن رمز الأقواس قد تضمنته نتائج مطابقة التعبير على النص.

و هذه هى طرق تخطى رموز ل Metacharacters فى التعابير النظامية.



يوجد فى التعابير النظامية اثنا عشر رمزا من رموز ل Metacharacters التى ينبغى تخطيها إذا كانت مستعملة فى التعبير النظامى بمعناها الحرفى، و هى:

- Backslash \ الخط المائل المقلوب
- Caret ^ علامة الإقحام
- Dollar sign $ علامة الدولار
- Dot . النقطة
- Pipe symbol | رمز الأنبوب
- Question mark ? علامة الاستفهام
- Asterisk * النجمة
- Plus sign + علامة الإضافة
- Opening parenthesis (قوس الاستفتاح
- Closing parenthesis) قوس الإغلاق
- Opening square bracket [قوس الاستفتاح المربع
- The opening curly brace { حاصرة الاستفتاح المجعدة

فى بعض الحالات ستبذل محركات (التعبير النظامى) أفضل ما عندها لفهم ما لو كانت هذه الرموز تحمل معنىً حرفيا أم لا، حتى فى حالة عدم تخطيها بإحدى الطرق الثلاث المذكورة سلفا من الأصل، فعلى سبيل المثال سيتم التعامل مع حاصرة الاستفتاح المجعدة } كرمز metacharacter فقط فى حالة ما إذا تبعه رقم يحدد عدد مرات التكرار كما ستتعلم لاحقا فى هذا الجزء.

*****

منظومات الرموز:

سنستخدم الـ Metacharacters للمرة الأولى لنتعلم كيف نستفيد من "منظومات الرموز".

و منظومات الرموز (كذلك تسمى بمجموعات الرموز): تتيح لنا تحديد الحرف (أو الرقم) المطلوب مطابقته في حالة ما إذا وافق أحد الحروف الواردة في منظومة الرموز التي حددناها.

و يتم تحديد منظومة من الرموز عن طريق استخدام رمز قوس الاستفتاح المربع] ثم يتبعه مجموعة الحروف المختارة ثم أخيرا نغلق المنظومة بقوس الإغلاق المربع [.

فعلى سبيل المثال دعونا نكتب تعبيرا نظاميا يمكننا بواسطته مطابقة كل من كلمتي affect و effect:

/[ae]ffect/

و كذلك يمكننا استعمال نطاق من الحروف (أو الأرقام) في منظومات الرموز، و يكون ذلك عن طريق وضع رمز الواصلة (-) بين حرفين (أو رقمين) ذوي صلة، كأن تكون المنظومة في التعبير النظامي وضعت لمطابقة أي حرف صغير محصور بين [a-z] فتكتب بالشكل السابق، كذلك أن تكون المنظومة قد وضعت لمطابقة أي رقم محصور في النطاق بين [0-9] فنكتبها في التعبير النظامي هكذا كما هي هنا.

و يمكن دمج نطاقات منظومات الرموز لتكون قادرة على مطابقة أي رمز في مقابلة عدة نطاقات، و ذلك بسلطة عن طريق وضع كل نطاق بعد الآخر، و لا يتطلب هذا الأمر الفصل بين النطاقات بطريقة معينة بل يمكن كتابتها متتابعة بلا فواصل. فعلى سبيل المثال إذا أردنا أن نطابق أي رمز حرفي-رقمي (سواء في حالته الصغيرة أو الكبيرة) في النص، فإنه يمكننا استخدام هذه المنظومة حينئذ: [0-9a-zA-Z] بدون الفصل بينها بمسافات،

و يمكن كذلك أن نكتب هذه المنظومة بدلا من تلك الطريقة عن طريق استخدام آلية الاتحاد [[0-9[a-z[A-Z]]] ، و معنى كل عنصر من عناصر المنظومة إذا كتبت بهذه الطريقة هو:

] طابق المجموعات التالية من الرموز

0-9 طابق أي شئ بين 0 و 9 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

أو

a-z طابق أي شئ بين a و z مثل (a, b, c, d, ..., z)

أو

A-Z طابق أي شئ بين A و Z مثل (A, B, C, D, ..., Z)

[قم بإنهاء منظومة الرموز

كذلك هناك إمكانية أخرى لكتابة المنظومة ذات النطاقات و تكون عن طريقة كتابتها بواسطة النطاقات السلبية أو نقيض النطاقات

حيث يمكننا عكس معنى منظومة الرموز بواسطة وضع رمز علامة الإقحام (^) في النطاق بعد رمز قوس الاستفتاح المربع ([).

فإذا كان لدينا منظومة رموز تحتوي على هذا النطاق [0-9] الذي يعني أي قيمة رقمية، فإن المنظومة السلبية لهذه المنظومة السابقة و هي التي تحتوي على النطاق السلبي لذلك المذكور سلفا و هو [^0-9] ستطابق أي رمز قيمته ليست قيمة رقمية.

من المفيد على كل حال ملاحظة أن هناك رموز موجودة بالفعل و لا تحمل قيما رقمية، فعلى سبيل المثال التعبير النظامى /hello[^0-9]/ لن يطابق سلسلة الحروف hello لانه ينبغى أن يتبعها رمز آخر و لا يحمل هذا الرمز قيمة رقمية كما جاء فى التعبير.

منظومات الرموز المعرفة سلفا:

بعد استعمال منظومات الرموز لبعض الوقت ستجد أن بعضها فى غاية الفائدة و النفع و أنه من الأفضل أن تكون لذلك البعض صيغة مختصرة ثابتة يتيسر استحضارها و استعمالها فى التعبير النظامى.

و لحسن الحظ فإن هناك عددا من منظومات الرموز المشهورة المعرفة سلفا و التى يمكن إعادة استعمالها فى التعبير النظامى مما يجعله سهل القراءة و الفهم.

و لا تقتصر منفعة هذه الرموز على كونها مختصرات معروفة و مشهورة لمنظومات رموز معينة، و لكنها كذلك يمكن أن تحمل معان مختلفة فى سياقات مختلفة.

فمثلا منظومة الرموز \w و التى تطابق أى رمز حرفى-رقمى، ستطابق كذلك أى منظومة رموز أخرى اعتمادا على (المعرف المحلى المعدل) و دعم (نظام الحروف الدولى الموحد Unicode).

أما عن منظومات الرموز المدعومة بالفعل فى بيثون فهى كالتالى:

. يطابق هذا العنصر أى رمز عدا رمز السطر الجديد \n

\d و يطابق هذا أى رقم عشرى، فهو مكافئ للمنظومة [0-9]

\D و يطابق هذا أى رمز غير رقمى، فهو مكافئ للمنظومة [^0-9]

\s و يطابق هذا رمز المسافة الفارغة، فهو مكافئ للمنظومة [\t\n\r\f\v]

\S و يطابق هذا أى رمز عدا رمز المسافة الفارغة، فهو مطابق للمنظومة [^\t\n\r\f\v]

\w و هذا يطابق أى رمز حرفى-رقمى، و على هذا فهو المكافئ للمنظومة [a-zA-Z0-9_]

\W و هذا يطابق أى رمز عدا الرمز الحرفى-رقمى، لذا فهو المكافئ للمنظومة [^a-zA-Z0-9_]

ملاحظة:

يتطلب العنصر الأول من المنظومات السابقة و هو النقطة (.)

المزيد من الانتباه، فهذه النقطة تعتبر تقريبا واحدة من أقدم رموز الـ metacharacters و أكثرها استخداما، و يمكنها مطابقة أى رمز من الرموز عدا رمز السطر الجديد.

و من المحتمل أن يكون سبب عدم مطابقتها لرمز السطر الجديد هو UNIX.

فأدوات سطر الأوامر "command-line" فى UNIX تعمل دائما سطرا تلو سطر "line by line"، و التعابير النظامية المتوفرة فى وقتنا هذا تطبق بصورة منفصلة على هذه الأسطر، و لهذا لا توجد رموز للسطر الجديد حتى تطابقها.

دعونا نضع رمز النقطة (.) فى تطبيق عملى بإنشاء تعبير نظامى يطابق ثلاثة من الرموز تحمل أى قيمة عدا رمز السطر الجديد.

/.../

شرح معانى عناصر التعبير النظامى السابق:

. يطابق أى رمز

. يطابق أى رمز يكون متبوعا بالرمز السابق

. يطابق أى رمز يكون متبوعا بالرمز السابق

و يعتبر رمز النقطة رمزا قويا جدا من رموز الmetacharacter و الذى يمكن أن يسبب مشاكل إن لم يتم استخدامه باعتدال.

فى كتابة التعبير النظامى يمكن اعتبار استخدام رمز النقطة فى معظم الحالات التى يستخدم فيها مظهرا من مظاهر الإسراف أو ربما يمكن القول أنه عارض من عوارض الخمول و الكسل.

و يعتبر استعمال منظومات الرموز أفضل و أسرع طريقة لاستعمال التعابير النظامية فى أداء مهام المطابقة على نص من النصوص بصورة دقيقة و بكفاءة عالية.

فعند العمل على امتدادات الملف (file paths) فى بيئة Windows أو Unix على سبيل المثال، يمكنك مطابقة أى رمز عدا الخط المائل و الخط المائل المقلوب باستعمال هذه المنظومة السالبة:

[^\/\] و هذه شرح معانى عناصرها:

[يطابق أى مجموعة من الرموز

^ لا تطابق الرموز التالية لهذا الرمز

\/ يطابق أى رمز من هذا النوع

\ يطابق أى رمز من هذا النوع

] نهاية منظومة الرموز

تخبرك منظومة الرموز هذه بصراحة أننا نقصد مطابقة أى رمز عدا رمز الفاصلة فى امتداد الملف فى كل من بيئتى Windows و UNIX.

*****

التعاقب (Alternation):

لقد تعلمنا فى السابق كيف طابق رمزا واحدا من ضمن مظومة من الرموز، و الآن سنتعلم طريقة أوسع و هى كيفية عمل المطابقة بواسطة مظومة من التعبيرات النظامية، و هو الأمر الذى يمكن إنجازه باستعمل رمز الأنبوب (|).

دعونا نفترض أننا نريد مطابقة أحد هاتين الكلمتين Yes أو No

و باستخدام التعاقب يمكننا كتابة التعبير النظام بكل سهولة كالآتى:

/Yes|No/

و معنى عناصر التعبير النظامى السابق:

طابق أيا من مظومات الرموز التالية

Yes الرموز Y،e، و s

أو

No الرموز N و o

و من جهة أخرى إن أردنا عمل المطابقة لأكثر من قيمتين فإنه يمكننا بكل بساطة الاستمرار فى إضافة القيم باستخدام رمز التعاقب (|) كالتالى :

/Yes|No|Maybe/

و ترجمة معانى عناصر التعبير السابق هى:

طابق أيا من مظومات الرموز التالية

Yes المظومة الحرفية "Yes"

أو

No المظومة الحرفية "No"

أو

Maybe المظومة الحرفية "Maybe"

عند استعمال التعابير النظامية الأكبر فمن المحتمل أن نحتاج إلى تغليف التعاقب "alternation" بداخل علامات حصر "parenthes" للإخبار بأن هذا الجزء فقط هو الذى يتم إجراء التعاقب عليه و ليس كامل التعبير النظامى.

فعلى سبيل المثال إن أخطأنا بعدم استعمل علامات الحصر () كما فى التعبير النظامى التالى:

/Licence: yes|no/

سيكون المعنى هكذا :

طابق أيا من مظومات الرموز التالية

Licence: yes الرموز s و ,L, i, c, e, n, c, e, :, ⇢, y, e

أو

no الرموز o و n

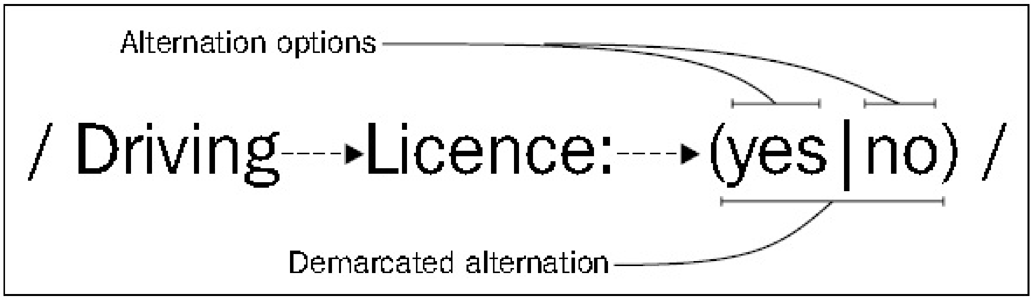
و بينما يعتقد البعض ممن كتب التعبير السابق بهذه الصورة أنه بالفعل يقوم بمطابقة:

Licence: yes أو Licence: no

إلا أن الواقع بالفعل هو أنه إنما يقوم بمطابقة:

Licence: yes أو no و ذلك لأن التعاقب تم تطبيقه على التعبير النظامي ككل و ليس فقط على الجزء المراد و هو yes|no

و لكن إذا أردنا أن نكتب التعبير السابق بطريقة صحيحة لتحقيق هدفنا، فلابد أن نكتبه بهذه الصورة:



******

Quantifiers (محددات الكمية):

لقد تعلمنا حتى الآن كيفية تحديد رمز مفرد فى تشكيلات متنوعة، و الآن سندعم عملنا السابق بما يسمى (Quantifiers) أو محددات الكمية، و هى عبارة عن آلية يتم بها تحديد كيفية تكرار رمز ما أو مظومة رموز أو رمز من رموز ال metacharacter.

فعلى سبيل المثال لو أردنا أن نصنع نموذجا للتحقق من عدد عناصر خانة فى عربة تسوق فيمكننا ذلك عن طريق تحديد مرات تكرار \d (مع الأخذ فى الاعتبار أن \d تطابق أى عدد عشرى).

على كل حل دعونا نبدأ بتعلم محددات الكمية الثلاثة الأساسية:

علامة الاستفهام ?

علامة الإضافة +

علامة النجمة *

و جميعها يحدد كمية (أو تكرار) الرمز السابق فقط

و معانيها بالترتيب كالآتى:

? احتمال تكرار الرمز السابق لها 0 أو 1 من المرات

* احتمال تكرار الرمز السابق لها 0 أو أكثر من المرات

+ احتمال تكرار الرمز السابق لها 1 أو أكثر من المرات

و هناك كذلك هذا المحدد {n,m} و يعنى أن احتمال تكرار الرمز السابق لها يقع بين n و m من المرات.

و لنأخذ مثالا تطبيقيا على لستعمل علامة الاستفهام ? كمحدد كمية فى التعبير التالى

```
/cars?/
```

و يمكن شرح التعبير كالتالى:

car عنصر يطابق c و a و r

s? عنصر يطابق حرف s بصورة اختيارية 0 أو 1 من مرات التكرار

و بناء على هذا فإن التعبير النظامى السابق يطابق كلا من الكلمة car و التى احتوت على 0 مرات تكرار للحرف s و كذلك يطابق الكلمة cars و التى احتوت على 1 مرات تكرار للحرف s.

الآن نحن نعلم كيف ندعم باستخدام محدد الكمية مظومات الرموز لتقبل رموزا مختلفة، و لكن هل من الممكن تطبيق محدد كمية على مظومة رموز؟

الجواب هو نعم، يمكن لمحددات الكمية أن تطبق على الرموز، منظومات الرموز، و حتى على المجموعات.

فمثلا يمكننا إنشاء تعبير نظامى مثل ذلك التالى للتحقق من أرقام الهواتف فى أى شكل من أشكالها المختلفة:

123 456 789

أو

123456789

أو

123-456-789

فنستعمل لذلك التعبير التالى:

/\d+[-\s]?\d+[-\s]?\d+/

و ترجمة عناصر التعبير تكون كالتالى:

\d منظومة رموز معرفة سلفا **معناها:** أى رمز عشرى

+ محدد كمية **معناه:** - مكرر مرة واحدة أو أكثر

[-\s] منظومة رموز **معناها:** أى شرطة أو رمز المسافة الفارغة

? محدد كمية **معناه:** - من المحتمل أن يظهر و من المحتمل ألا يظهر

\d منظومة رموز معرفة سلفا **معناها:** أى رمز عشرى

+ محدد كمية **معناه:**- مكرر مرة واحدة أو أكثر

[-\s] منظومة رموز **معناها:** أى شرطة أو رمز المسافة الفارغة

? محدد كمية **معناه:** - من المحتمل أن يظهر و من المحتمل ألا يظهر

\d منظومة رموز معرفة سلفا **معناها:**- أى رمز عشرى

+ محدد كمية **معناه:** - مكرر مرة واحدة أو أكثر

ذكرنا فى بداية هذا القسم نوعا آخرا من محددات الكمية و التى تستعمل الحاصرات المجنحة "curly braces" و هذا شكلها ({})، و باستعمال هذا التركيب يمكننا تحديد عدد مرات ظهور الرمز الذى يسبقه فى التعبير النظامى بدقة تامة.

فمثلا إذا استعملنا {5} بعد آخر رمز فمعنى هذا أن عدد مرات ظهوره هى خمس مرات.

و بالتالى:

فالتعبير النظامى \w{8} يحدد أن عدد مرات ظهور الرمز الحرف-رقمى هى 8 مرات بالتحديد.

كذلك فإنه يمكننا تحديد نطاق معين للتكرار عن طريق تحديد الحد الأدنى و الحد الأعلى لعدد مرات التكرار.

فمثلا إن كان نطاق التكرار محصورا بين 4 مرات و 7 مرات فإنه يمكننا تعريفه بواسطة التركيب {4, 7} .

يمكن أن تحذف كل من القيمة الصغرى أو الكبرى إهمالا ل0 و ما لا يحصى على التوالي.

فلتعيين تكرار حتى 3 مرات يمكننا استعمل {,3}، و يمكننا كذلك لتعيين تكرار 3 مرات على الأقل يمكننا استعمل {3,} .

لدينا أربع مجموعات مختلفة لهذا النوع يبينها الجدول التالي:

التركيب	الوصف
{n}	الرمز السابق يتكرر بالتحديد n من المرات
{,n}	الرمز السابق يتكرر على الأقل n من المرات
{n,}	الرمز السابق يتكرر على الأكثر n من المرات
{n,m}	الرمز السابق يتكرر ما بين n و m من المرات (متضمنا كليهما)

أنشأنا سابقا في هذا الكتاب تعبيرا نظاميا للتحقق من أرقام الهواتف وشرحنا معنى كل عنصر من عناصر التركيب في السابق.

دعونا الآن نعمل على نفس التركيب لتطبيق هذه الآلية المذكورة سلفا في تحديد عدد مرات التكرار:

/ \ d{1,3}[- \ s]?\ d{3}[- \ s]?\ d{3}/

و يكون معنى مجموعات الحواصر المجعدة كالتالي:

المجموعة الأولى في أقصى اليسار معناها أن المجموعة الرقمية يمكن أن تحتوي على ما قد يصل إلى 3 رموز، بينما المجموعتان الباقيتان يشيران إلى أن ما يسبقهما من المجموعات الرقمية لابد أن يحتويا على ثلاثة أرقام فقط.

(الطماع و الزاهد):

تسلك محددات الكمية سلوكين متناقضين عند عمل المطابقة على نص ما، فالسلوك الأول و هو السلوك الافتراضي للمحددات يسمى (الطماع) و فيه يحاول المحدد أن يطابق أكبر كمية ممكنة من النص للحصول على أضخم نتيجة مطابقة ممكنة.

أما السلوك الثاني للمحددات فهو على النقيض من ذلك و يسمى (الزاهد) حيث يعمل المحدد فيه على الحصول على أقل نتيجة مطابقة ممكنة، و يتم تهيئة محدد الكمية ليسلك هذا المسلك أثناء المطابقة عن طريق إضافة علامة استفهام (?) إضافية إلى المحدد كالتالي: ?? , ?* , ?+.

مطابقات الحدود (Boundary Matchers):

قمنا حتى الآن باستكشاف سلوك التعابير النظامية عند إجراء المطابقة بداخل نص ما. و في بعض الأوقات قد نحتاج إلى مطابقة "سطر كامل" و هو ما يتطلب مطابقة بداية السطر أو حتى آخره، و هو ما يمكن عمله بواسطة (مطابقات الحدود).

فمطابقات الحدود هى عدد من أدوات التعريف التى تطابق

موضعا معينا فى المدخلات عند تطبيقها عليها.

و إليك مطابقات الحدود المتوافرة فى لغة بيثون:

^ و تطابق بداية السطر

$ و تطابق نهاية السطر

\b و تطابق حد الكلمة

\B و هى عكس السابقة، حيث تطابق أى شئ ليس حدا للكلمة

\A و تطابق بداية المدخلات

\Z و تطابق نهاية المدخلات

ستتصرف مطابقات الحدود هذه بصورة مختلفة باختلاف السياق. فعلى سبيل المثال سيعتمد مطابق الحدود (\b) مباشرة على "المعرف المحلى المعدل" لتحديد حدود الكلمة حيث أن حدود الكلمة تختلف من لغة إلى أخرى.

كما أن مطابقات حدود بداية و نهاية السطر ستتصرف كذلك بصورة مختلفة اعتمادا على إشارات معينة سنعرفها لاحقا إن شاء الله عز و جل.

دعونا نأخذ مثالا لتوضيح طريقة عمل مطابقات الحدود:

إذا أردنا مثلا أن نكتب تعبيرا نظاميا لمطابقة الأسطر التى تبدأ بكلمة : "Adress:" فيمكننا كتابته كالتالى:

/^Adress:/

حيث معانى عناصر التعبير هى:

^ يطابق بداية السطر

A يطابق حرف A التالى له

d يطابق حرف d التالى له

r يطابق حرف r التالى له

e يطابق حرف e التالى له

s يطابق حرف s التالى له

s يطابق حرف s التالى له

: يطابق رمز النقطتين التالى له

و إذا أردنا أن نأخذ خطوة أبعد من ذلك فنستخدم علامة $ مع علامة ^ لمطابقة كل من بداية و نهاية السطر، فلابد أن نضع فى الحسبان أننا فى هذه الحالة سنطابق السطر ككل و ليس فقط نموذجا متضمنا فى السطر.

و تعقيبا على المثال السابق، دعونا نجرب مطابقة ما بعد كلمة Adress للتأكد من كونها حتى نهاية السطر رموزا حرفية أو مسافات فارغة.

و لعمل مطابقة للسطر ككل حتى نهايته فسنضع مظومة رموز تتوافق مع الرموز المطلوبة و نحدد كمية تكرارها بأى عدد من المرات حتى نهاية السطر و هو ما يعبر عنه التعبير النظامى التالى:

```
/^Name:[\sa-zA-Z]+$/
```

و هذا شرح عناصر التعبير بالتفصيل:

^ يطابق بداية السطر

A يطابق حرف A التالى له

d يطابق حرف d التالى له

r يطابق حرف r التالى له

e يطابق حرف e التالى له

s يطابق حرف s التالى له

s يطابق حرف s التالى له

: يطابق رمز النقطتين التالى له

[\sa-zA-Z] ثم يطابق أى مسافة فارغة تابعة له أو أى رمز حرفى صغير أو كبير

+ و هذا الرمز السابق يمكن أن يكرر مرة واحدة او أكثر

$ حتى نهاية السطر

هناك مطابق حدود آخر مهم و هو "حد الكلمة" \b، و الذى سيطابق أى رمز لا يكون رمز كلمة، و من ثم أى حد محتمل للكلمة.

و هذا أمر فى غاية النفع خاصة عندما نريد العمل مع الكلمات المفردة و لا نريد إنشاء مظومات رموز بكل رمز مفرد من المحتمل أن يقسم كلماتنا (كالمسافات، الفواصل، النقاط، الشرطات، إلخ). فيمكننا على سبيل المثال التأكد من أن كلمة Adress قد وردت فى نص ما عن طريق استخدام التعبير النظامى التالى:

```
/\bAdress\b/
```

و معنى عناصره كالتالى:

العنصر	الوصف

\b يطابق حد كلمة

A يطابق حرف A التالى له

d يطابق حرف d التالى له

r يطابق حرف r التالى له

e يطابق حرف e التالى له

s يطابق حرف s التالى له

s يطابق حرف s التالى له

\b ثم يطابق حد كلمة آخر تالى له

و استخدام التعبير النظامى السابق يكون أفضل فى حالة إذا ما أردنا مطابقة كلمة Adress مفردة وحدها بتمامها دون أى إضافة، و هى النتيجة التى لا تتوفر إذا استعملنا التعبير النظامى \Adress\ و الذى سيطابق جميع الكلمات المحتوية على كلمة Adress مثل Adressed و Adresses.

ملخص:

فى الجزء الأول تعلمنا أهمية التعابير النظامية، و كيف قد أنها أصبحت أداة عظيمة الصلة بالمبرمجين.

كذلك درسنا التعابير النظامية بصورة نظرية و عامة و تعلمنا التراكيب الأساسية لها و بعض معالمها الرئيسية مثل مظومات الرموز و محددات الكمية.

سنتعلم فى الجزء التالى إن شاء الله عز و جل كيف يمكن استعمال التراكيب النظامية فى لغة بيثون عن طريق استعمال (re module).

*****